CSC 406 – Computer Graphics

Programming Assignment 01, Fall 2011

Monday, September 26, 2011 [fixed Tuesday, Sept. 27]

Due date: Monday, October 04

1 What this Assignment is About

1.1 Objectives

In this assignment, we have three objectives:

- To write a first small OpenGL (or jogl) program;
- To get used to the translation, rotation, and scaling transformations of OpenGL;
- To use OpenGL to perform matrix computations, or perform matrix computations and apply them in OpenGL.

This is an individual assignment. You are of course encouraged to discuss it with other students but I expect you complete it by yourself.

1.2 Handouts

The handouts for this assignment consist of

- this document, as a pdf file;
- a small code fragment for displaying the local reference frame.

1.3 GUI setup

For this program you simply need to create a square viewport (in OpenGL, create a square window; in jogl, it's the panel in which you render that should be square). No need for any menu elements our mouse event handling. Our entire interface will be keyboard-based.

1.3.1 "Special keys" in glut

In glut, "special" keys such as arrow keys are not handled by the `glutKeyboardFunc` callback but by the `glutSpecialFunc` callback. Otherwise the handling of "special" key events is quite similar to that of regular key events.

```
void mySpecialKeyFunc(int key, int x, int y) {
    switch (key) {
        // case of left arrow
        case GLUT_KEY_LEFT:
            ...your code goes here
            break;

        // case of right arrow
        case GLUT_KEY_RIGHT:
            ...your code goes here
            break;

        // case of up arrow
        case GLUT_KEY_UP:
            ...your code goes here
            break;

        // case of down arrow
        case GLUT_KEY_DOWN:
            ...your code goes here
            break;
    }

    // probably needed here: force display
    glutPostRedisplay();
}
```

Of course, you need to set up the call back when you create your window:

```
glutSpecialFunc(mySpecialKeyFunc);
```

1.3.2 Special keys in jogl

In jogl all the event handling is done by the JVM, so you handle arrow and other "special" keys the way you would handle any other key strokes in any other Java application.

```
public void keyPressed(KeyEvent e) {
    switch (e.getKeyCode()) {
        // case of left arrow
        case KeyEvent.VK_LEFT:
            ...your code goes here
            break;

        // case of right arrow
        case KeyEvent.VK_RIGHT:
            ...your code goes here
```

```
                break;

            // case of up arrow
            case KeyEvent.VK_UP:
                ...your code goes here
                break;

            // case of down arrow
            case KeyEvent.VK_DOWN:
                ...your code goes here
                break;
        }

        // probably needed here
        repaint();
    }
```

1.4 World dimensions

The "world" you will be drawing into is square and covers the range $[-10, 10]$ for both x and y.

2 What to Do

2.1 Application behavior

Your application must at any time display the current reference frame (resulting from a series of transformations applied by the user). The user will be able, through simple key strokes:

- to change the current reference frame (translation, rotation, scaling, skewing);
- to draw a simple shape at the location of the current reference frame;
- to return to a previously stored reference frame.

2.2 Basic keystrokes

2.2.1 Mode selection

- 'r' or 'R' selects the frame rotation mode;
- 't' or 'T' selects the frame translation mode;
- 's' or 'S' selects the frame scaling mode.

2.2.2 Role of the arrow keys

- In frame rotation mode, the left and right arrow keys rotate the frame counterclockwise and clockwise respectively, by increments of 15 degree;
- In frame translation mode, the left and right arrow keys control frame translation along the **local** x axis while the up and down arrow keys controls frame translation along the **local** y axis, by steps of 0.25 in the current frame;
- In frame scaling mode, the left and right arrow keys control frame scaling along the **local** x axis while the up and down arrow keys controls frame scaling along the **local** y axis. Frame scaling consists in scaling up/down along the chosen direction by a factor of 1.2.

2.2.3 Action keys

- 'q' or 'Q' ends execution;
- the space key is used to add to the scene a small square of side 1 drawn in the local reference frame.

2.2.4 Frame storage

Your application will provide "storage" for 4 reference frames, $\mathcal{R}_a$, $\mathcal{R}_b$, $\mathcal{R}_c$, $\mathcal{R}_d$. These frames are all initialized to coincide with the camera /world reference frame.

- 'A' saves the current reference frame as $\mathcal{R}_a$;
- 'B' saves the current reference frame as $\mathcal{R}_b$;
- 'C' saves the current reference frame as $\mathcal{R}_c$;
- 'D' saves the current reference frame as $\mathcal{R}_d$.

and then

- 'a' sets the current reference frame to coincide with $\mathcal{R}_a$;
- 'b' sets the current reference frame to coincide with $\mathcal{R}_b$;
- 'c' sets the current reference frame to coincide with $\mathcal{R}_c$;
- 'd' sets the current reference frame to coincide with $\mathcal{R}_d$.

2.3 Number of objects to draw

Your application should allow the user to draw 10 objects (corresponding to 10 different reference frames). After 10 objects have been created, a newly created object will simply replace the oldest object on the list.

2.4 Two versions of the code

You must provide two versions of the application. One in which all geometric transformations are performed using the OpenGL `glTranslate`, `glRotate`, and `glScale` functions, and the other versions in which you perform these operations exclusively by using the `glSetMatrix` and `glMultMatrix` functions.

3 Various Extra Credit (for a maximum of 130 pts)

3.1 Allow an unlimited number of objects (10 pts)

To get full credit here, you must use either a vector or a list, in C++, or a Vector or an ArrayList, in Java. Resizing a regular array will only give you 3 points of extra credit. Using a ridiculously large array will not give you any extra point (and quite frankly should even cost you some!).

Note: In the next assignment you will *have* to handle lists of objects, so looking into the STL `vector` and `list` classes (in C++) or `Vector` and `ArrayList` (in Java) is definitely time well spent, even if you can't complete this extra credit section.

3.2 Print vertex coordinates of a new shape (up to 10 pts)

When the user hits the space bar to create a new shape, print to the console the coordinates of the four vertices of the new shape in the original reference frame.

You get 5 points for each of the two versions of application (matrix-based vs. operation-based) for which you add this feature.

3.3 Implement skewing (up to 15 pts)

For this we need to define new key behaviors.

3.3.1 Mode selection

- 'k' or 'K' selects the frame skewing mode.

3.3.2 Role of the arrow keys

- In frame skewing mode, the left and right arrow keys control the "amount of x" added to the y axis, by increments of 0.1;
- In frame skewing mode, the up and down arrow keys control the "amount of y" added to the x axis, by increments of 0.1.

3.3.3 Extra points breakup

You get 10 pts of extra credit for adding skewing to the "matrix-based" version of your application. You would get an extra 5 pts of extra credit for adding it to the "operation-based" version, but quite frankly I can't think of any non-ugly way of doing that.

3.4 Translation relative to the "old"/original frame (10 pts)

You should only add this feature to the "matrix-based" version of the application. For this new feature we need to define new key behaviors.

3.4.1 Mode selection

- 'o' or 'O' selects the mode for translating the current frame along the axes of the old/original reference frame.

3.4.2 Role of the arrow keys

- the left and right arrow keys control frame translation along the **original** x axis;
- the up and down arrow keys control frame translation along the **original** y axis.

4 What to hand in

4.1 Documentation

4.1.1 Inline comments

You should properly comment and document your code. This means that after cleaning up your code (including the one you use from the handout), you should provide inline comments to explain what is going on there[1]. I remind you that code-level comments are not suppose to paraphrase in plain English the code that just follows it. So, the following comment is useless/ridiculous:

```
// increment x by one
x++;
```

The purpose of code-level comments is to explain what a small block of code does, for example by giving the outline of the algorithm or by re-stating the code's objectives.
The level of details of comments should be adapted to the level of your intended reader. Here at the 406-level, I don't need you to give me line-by-line comments for some elementary blocks of code. For example, "find largest element in array a" or, when multiplying two matrices "C = A . B" is good enough for me.

[1] I prefer inline comments as opposed to block comments because you can't nest block comments.

4.2 Javadoc-style comments for functions

Next week we will start writing graphic classes and C++ programmers will have to provide header files for these classes, and javadoc-style comments for these classes. Just to get prepared for that, I want you to start writing Javadoc-style comments for all the functions in your code (I only really care about the function summary, detailed explanation, and `@param`, `@return`, and possibly `@see` fields).

Java programmers can naturally run javadoc on their code to produce the documentation. C++ programmers can use Doxygen (a free download) on their source file (starting with next assignment we will only use header files to produce the documentation).

4.3 Source code

Provide all the source files and header files needed to compile and run your program. Remember that your code should be able to compile and execute "as-is" with Eclipse, so don't include any platform-specific headers in your code. If the file `glPlatform.h` provided in the handout does not work for you, get in touch with the TA and me ASAP.

4.4 Report

For this assignment the report consists of a simple user manual giving the list of commands available to the user. Definitly mention if you made changes to the specifications for the assignment (and explain why) and which extra credit features (if any) you implemented. Also discuss possible difficulties encountered while completing the assignment.

5 Grading

If you submit code that could not possibly compile (contains obvious syntax errors) your assignment will not be graded and you will get a grade of 0. For all others:

- the code does what it is supposed to do: 45%
- comments: 15%
- readability of the code: 20%
- proper identifier names: 10%
- report: 10%